PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G07C 1/22, A63B 26/00

A1

(11) International Publication Number: WO 92/21106

(43) International Publication Date: 26 November 1992 (26.11.92)

(21) International Application Number: PCT/AU92/00237

(22) International Filing Date: 22 May 1992 (22.05.92)

(30) Priority data:
PK 6276 22 May 1991 (22.05.91) AU

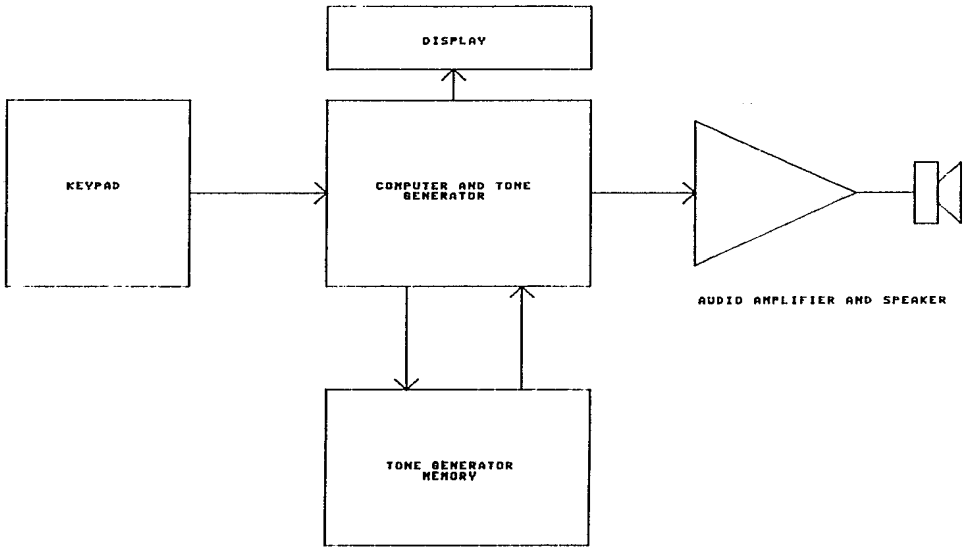
(71)(72) Applicant and Inventor: DAWSON, Ronald, Gerry [AU/AU]; 35 Tareena Street, Nedlands, WA 6009 (AU).

(74) Agent: KILDEA, Paul, F.; P.O. Box 4577, Kingston, ACT 2604 (AU).

(81) Designated States: AT (European patent), AU, BE (European patent), CA, CH (European patent), DE (European patent), DK (European patent), ES (European patent), FR (European patent), GB (European patent), GR (European patent), IT (European patent), JP, KR, LU (European patent), MC (European patent), NL (European patent), SE (European patent), US.

Published
With international search report.

(54) Title: SPORTS TRAINING DEVICE



(57) Abstract

A sports training device provides synchronisation signals to induce and guide movements of a sportsperson engaged in a sporting activity. The device comprises a digital logic computer and a tone generator, the computer logic being programmed to activate the tone generator in accordance with stimulus parameters. Means are provided to input into the computer predetermined stimulus parameters based upon a behavioural analysis of models of relevant movement sequences of the sporting activity to cause the tone generator to generate a sequence of auditory pulses having predetermined characteristics. These characteristics such as intensity, duration, quality and the like relate to movements of different parts of the body and/or provide other information concerning the particular movement. The device also includes audio output means through which the generated sounds are relayed to the sportsperson as a preview and guide to the sporting activity.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	**FI**	Finland	**ML**	Mali
AU	Australia	**FR**	France	**MN**	Mongolia
BB	Barbados	**GA**	Gabon	**MR**	Mauritania
BE	Belgium	**GB**	United Kingdom	**MW**	Malawi
BF	Burkina Faso	**GN**	Guinea	**NL**	Netherlands
BG	Bulgaria	**GR**	Greece	**NO**	Norway
BJ	Benin	**HU**	Hungary	**PL**	Poland
BR	Brazil	**IE**	Ireland	**RO**	Romania
CA	Canada	**IT**	Italy	**RU**	Russian Federation
CF	Central African Republic	**JP**	Japan	**SD**	Sudan
CG	Congo	**KP**	Democratic People's Republic of Korea	**SE**	Sweden
CH	Switzerland			**SN**	Senegal
CI	Côte d'Ivoire	**KR**	Republic of Korea	**SU**	Soviet Union
CM	Cameroon	**LI**	Liechtenstein	**TD**	Chad
CS	Czechoslovakia	**LK**	Sri Lanka	**TG**	Togo
DE	Germany	**LU**	Luxembourg	**US**	United States of America
DK	Denmark	**MC**	Monaco		
ES	Spain	**MG**	Madagascar		

SPORTS TRAINING DEVICE

Field of Invention

The invention relates to training a sportsperson for some sporting activity and, in particular, to regulating the movements of the sportsperson engaged in that activity. The invention has application to all kinds of sports and to a wide range of sportspersons including a novice commencing to learn the rudiments of some sport as well as someone more proficient seeking to improve performance.

Background of Invention

Although related to the particular activity, sports training has, generally speaking, followed a miscellany of procedures varying from the casual to the systematic. The latter category has involved the employment of coaches, special training facilities and an assortment of sophisticated equipment. Timing an activity has been commoplace. The invention concentrates on temporal intervals between specific movements and on temporal proportionality of complex movements.

Description of the Invention

Broadly, in accordance with the invention, predetermined signals are used as timing synchronizers to induce and guide the execution of movements by a sportsperson engaged in a particular sporting activity. The signals may be produced by an electronic device which has been programmed to generate a

sequence of auditory pulses having predetermined characteristics.

In the first place, the predetermined signals are dependent upon the particular sporting activity. In addition, characteristics of the sequential pulses are derived from a prior analysis of the movements involved in an appropriate sporting behaviour. The prior analysis may be based upon an optimum model of performance where the sportsperson is being trained to achieve an output for which there is an accepted standard. Alternatively, the prior analysis may be based upon a model derived from a study of the sportsperson's own behaviour. For example, the derived model may be used repetitively by that particular sportsperson in order to achieve consistency in timing. In another example, a number of models may be analysed so that the sportsperson may experiment with different timing strategies. The model may be a human one or it may be mechanical.

From the analysis of the appropriate sporting behaviour, stimulus parameters are derived. These parameters may include the onset of movement of a body part, the duration of movement and the relative timing of movements of different parts of the body. Other information such as speed or force of movement may be included. The stimulus parameters are used to vary characteristics of the auditory pulses such as intensity or duration or quality or the like to relate to movements of different parts of the body and/or to provide other information concerning the particular movement.

The stimulus parameters are fed into a programmable computer by such means as a keyboard. The auditory pulses are relayed to the sportsperson by audio output means. Preferably, the audio output means include an individual earpiece for each sportsperson. The audio output means may include a radio link to a remote sportsperson. In either case, timing information may be relayed directly and instantaneously to a sportsperson.

In accordance with the invention, a sports training device to provide synchronisation signals to induce and guide movements of a sportsperson engaged in a sporting activity comprises a digital logic computer and a tone generator, the computer logic being programmed to activate the tone generator in accordance with stimulus parameters, means to input into the computer predetermined stimulus parameters based upon a behavioural analysis of models of relevant movement sequences of the sporting activity to cause the tone generator to generate a sequence of auditory pulses having predetermined characteristics and audio output means through which the generated sounds are relayed to the sportsperson as a preview and guide to the sporting activity.

Brief Description of the Drawings

Fig.1 is a block diagram illustrating a sports training device in accordance with the invention; and

Figs.2a and 2b combined show a circuit diagram illustrating one embodiment of a sports training device in accordance with

the invention.

Detailed Description of the Embodiments

To illustrate the invention, two applications thereof will be discussed. For convenience, these applications will be identified as "sports-synch" and "sports-pacer", respectively. A single sports training device may be designed so as to be suitable for use in both applications. Alternatively, separate devices may be designed specifically for one or other application.

The sports-synch is intended primarily for what may be described as discrete activities such as hitting a golf ball or hitting a cricket ball. Such activities can be made more precise if the onset and duration of various body movements can be signalled precisely to the sportsperson. Incidentally, those two particularised activities illustrate the flexibility of the sports-synch to control self-contained, internally triggered actions as occur in golf as well as actions which have external timing requirements as in cricket.

The ideal golf swing involves synchronous movement of several parts of the body. A sequential signal pattern may be based on an analysis of the golf swing using a human or mechanical model. The synchrony may be signalled to the golfer by a sequence of different auditory tones which signal the onset of movement for different body parts. Preferably, the whole sequence commences with a brief synchronous tone burst at, for example, 2 per second. This tone burst acts as an onset

signal and may be triggered at the golfer's discretion. The golfer will learn which tones are the trigger for movement of particular parts of the body and will then practise to put the sequence together with the timing indicated. Thus, the golfer will learn to maintain consistency in timing.

The sports-synch also has application to cricket batting strokes. Efficient stroke-play in cricket involves a multiplicity of decisions in a short space of time. Sports-synch will enable a batsman to practise the timing of specifically identified shots. The timing sequence of the shot could be based upon an ideal model or, alternatively, on an individual model as, for example, in the case of juniors whose body proportions do not allow them to approach the ideal.

An analysis of skilled ball hitting (e.g., Bootsma and Wieringen, 1988) indicates that the external trigger for ball hitting is consistently related to the distance the ball is from the eyes. For this reason, the onset of the timing signals generated by a sports-synch for cricket shots will preferably be based upon an analysis of a practised professional playing against a conventional bowling machine. Thus, unlike the sports-synch for golf, the device for cricket is externally triggered. However, like the golfing version, the sports-synch for cricket will generate tones identifying the movement of the certain parts of the body. A batsman will learn which tones relate to particular body parts. He will then practise the shot, initially without a ball and then, ideally, with a bowling machine. Each shot will be identified

by a different tonal sequence and, preferably, a different onset signal for the commencement of each shot. The advantage of using the bowling machine is that the batsman could practise a certain shot over and over again provided that the bowling machine is set up to deliver a ball at constant length and velocity. In this event, the timing sequence would preferably be initiated remotely from the bowling machine by, for example, a radio link. If a human bowler were to be used, the timing sequence for the shot could be initiated remotely by a third party such as a coach. Thus the timing of each tone would be programmed in advance from a model such that the sequence for an activity may be triggered in full from a single input. The single input trigger could be initiated manually or by a remote signal from another device such as a bowling machine. The sports-synch could also be reprogrammed by the user in order to change various features of the total event to suit the individual.

The sports-synch may be used in other activities which culminate in a precise movement which has a timing prerequisite such as high-jumping, bowling a cricket ball, putting, etc. All of these activities involve a single sequence of events which should be tailor-made and then initiated singly.

On the other hand, the sports-pacer is intended for the timing of measured repetitive movements such as occurs in running and swimming. The essential purpose of this device is to deliver auditory signals which are to be synchronized with the mode of propulsion (e.g., a pace in running or a swimming

stroke) such that the pacing feature is immediately convertible into a measure of velocity. Thus, a sports-pacer acts as a speedometer for the athlete. The accuracy of the speedometer function is dependent upon measurements of the particular athlete performing over set distances so that paces or strokes per distance can be converted to pulses per unit time. Given this information, it is possible to programme a training regimen for an athlete or a full race without the athlete having constantly to check a time-piece.

Preferably, the device would be flexible enough to correct for changes in terrain, simply by the athlete or coach noting the change in distance travelled over changes in slope of running surface. Thus, a race like a marathon could be programmed from start to finish. The athlete, in full knowledge of his speed throughout the race, would be able to preset his pace for the race in advance.

An athlete would also be able to test out different strategies for racing given that the pacer would enable the athlete to race at different velocities at different stages of the race, with a precise knowledge of what those velocities are.

An application of the device which differs slightly from the prior examples is to aid in synchronizing the run-up of a bowler in cricket. Fast bowlers in particular need a precise rhythm when they bowl. A sports-pacer would be able to provide a series of pulses to pace each step in the run-up

and the subsequent arm movements leading to the delivery of the ball.

A sports-pacer needs to be a more flexible device than a sports-synch. The device would have inbuilt programmes of performance based, for example, upon the measurement of world-class athletes in appropriate races which can be used as a model. The device could also be based upon individual programmes over set distances.

The sports training device illustrated in Fig.1 is suitable for both the sports-synch and sports-pacer applications. The device comprises a programmable electronic system made up of three main components. The first is an input device with a keypad which is used to select programmes (if there is more than one programme) and input stimulus parameters. The second is a computer and tone generator with an associated memory made with programmable microchips. The third is an audio amplifier and speaker, through which the sounds generated by the computer are relayed to a sportsperson.

The device allows the user to select sounds covering a wide range of frequencies and intensities and arrange them in sequence. The sequence can then be played, on command from the keypad, through any of a number of speaker or earpiece outputs. The device may also contain a display which indicates to the user the precise details (frequency, duration, sequence, etc.) of the information currently programmed. The output characteristics of the device can

cover the whole range of audible frequencies of sound, the tonal durations may range from milliseconds to seconds and the total duration of the auditory sequences can be as short as milliseconds or as long as hours. The device may include means whereby a number of different auditory sequences can be stored concurrently. Further, the device may incorporate more than one programme. In this event, the appropriate programme and auditory sequence may be selected through operation of the keypad.

Preferably, the audio output means used to relay a sequence of auditory tones to a sportsperson comprises an earpiece which may be worn by the sportsperson. In the case where the same sequence of auditory tones is to be relayed to more than one sportsperson, individual earpieces may be supplied to each person. Where the sportsperson is remotely located as in a marathon, the audio output means should include a radio link.

To give greater portability, it is preferred that the training device be battery powered.

Figs. 2a and 2b depict a circuit diagram for a sports training device according to one embodiment of the invention. With this circuit and the computer programme hereinafter detailed, the training device may be used either as a sports-synch or a sports-pacer. The major differences in function result from the way in which tonal sequences are selected by the programme and are stored in the hardware and triggered by the

sportsperson. In the case of the sports-synch function, preparatory signals start off a sequence. These are followed by a series of tones, whose frequency and inter-pulse-intervals have been selected so as to guide a whole-body action involving the movement of many parts. In the case of the sports-pacer function, the range of tones used will be less extensive, since it is the repetitive feature of a particular movement which will be signalled; however, the output will be such as to cover the repetitive movement sequence for the total duration of a sporting activity, such as the running of a marathon, which takes over two hours.

The circuit shown in Figs. 2a and 2b represents a programmable tone sequence generator which is controlled by a Motorola (MC 68705C8) microcontroller UI. The controller UI monitors the input keys of keyboard KI and performs all timing and tone selection functions for the device operation in either of its sports-synch or sports-pacer applications. The tones and times are stored in the processor ROM and are accessed by the CPU to generate precisely controlled tones and accurate durations.

The circuit also includes a reset generator (MC34064) which monitors the power supply and holds controller UI in a reset condition during power failure or low battery voltage. Voltage regulator (MC78L05) U3 regulates the battery voltage to give +5 volts for the digital circuit. A generator U4 generates -5 volts from the +5 supply for the microchip of tone generator (ML2035) U5 which takes serial data from

controller U1 via the SPI in hexidecimal format to produce a sine wave. A low power amplifier (LM386) U6 takes sine wave from tone generator U5 and provides sufficient power to drive low impedance head phones or earpiece. The power is provided by a 9 volt battery B1.

For the operation of the training device in the manner described, a programme suitable for use with the circuit shown in Figs. 2a and 2b is as follows:-

```
;Programable tone sequence generator

;*****************************************************
;*        EQUATES                                    *
;*****************************************************

;*   I/O Ports

PA:       EQU  $00          ;Port A Data/address LO Bus
PB:       EQU  $01          ;Port B address HI Bus
PRC:      EQU  $02          ;Port C 0123 In, 4567 Out
PD:       EQU  $03          ;Port D Inputs only

;*   Data Direction Registers

DDRA:     EQU  $04          ;I/O Port
DDRB:     EQU  $05          ;Out Port
DDRC:     EQU  $06          ;In/Out Port

;*   Serial Periphial Interface Registers

SPCR:     EQU  $0A          ;Control Register
SPSR:     EQU  $0B          ;Status Register
SPDR:     EQU  $0C          ;I/O Data Register

;*   Serial Comms Interface Registers

BRR:      EQU  $0D          ;Baud Rate Register
SCCR1:    EQU  $0E          ;Control Reg 1
SCCR2:    EQU  $0F          ;Control Reg 2
SCSR:     EQU  $10          ;Status Reg
SCDAT:    EQU  $11          ;I/O Data Reg

;*   Timer Registers

TCR:      EQU  $12          ;Timer Control Reg
TSR:      EQU  $13          ;Timer Status Reg
ICRH:     EQU  $14          ;Input Capture Reg HI
ICRL:     EQU  $15          ;Input Capture Reg LO
TOCRH:    EQU  $16          ;Output Compare HI
TOCRL:    EQU  $17          ;Output Compare LO
TCRH:     EQU  $18          ;Timer Count HI
TCRL:     EQU  $19          ;Timer Count LO
TCARL:    EQU  $1A          ;Timer alternate Count HI
TCARH:    EQU  $1B          ;Timer alternate Count LO

;*   Timer Registers in RAM

TEMP1L:   EQU  $50          ;
TEMP1H:   EQU  $51          ;
TEMP2L:   EQU  $52          ;
TEMP2H:   EQU  $53          ;
MSEC:     EQU  $54          ;TISR 0.001 second counter
HSEC:     EQU  $55          ;TISR 0.1   second counter
NEWMSECL: EQU  $56          ;
NEWMSECH: EQU  $57          ;
```

```
;*    Ram Pointers

RAMDATA:   EQU   $58          ;RAM data IN/OUT
RAMADDL:   EQU   $59          ;
RAMADDH:   EQU   $5A          ;



RAMSTL:    EQU   $5B          ;
RAMSTH:    EQU   $5C          ;
RAMPNTL:   EQU   $5D          ;Next RAM address
RAMPNTH:   EQU   $5E          ;

TEMP:      EQU   $5F          ;Temp store
TEMPA:     EQU   $60          ;Temp store A
TEMPX:     EQU   $61          ;Temp store X
TEMPL:     EQU   $62          ;Temp store for ascon
TEMPM:     EQU   $63          ;Temp store for ascon
TONECNT:   EQU   $64          ;Tone counter
FLAG:      EQU   $65          ;Flag register
BEEPS:     EQU   $66          ;No. of notes in sequence
TIMOUT:    EQU   $67          ;Temp store
KEY:       EQU   $68          ;Key number
DPLYPNT:   EQU   $69          ;LCD position pointer
SCALE:     EQU   $6A          ;Storeit
SEQCNT:    EQU   $6B          ;Note sequence counter
TICL       EQU   $6C          ;TISR period counter lo
TICH       EQU   $6D          ;TISR period counter hi
MSBY:      EQU   $6E          ;
LSBY:      EQU   $6F          ;
NUMBER:    EQU   $70          ;
DATACNT:   EQU   $71          ;
DIGIT5:    EQU   $72          ;Tone duration millisecs
DIGIT4:    EQU   $73          ;Tone duration hundredths
DIGIT3:    EQU   $74          ;Tone duration tenths
DIGIT2:    EQU   $75          ;Tone duration seconds
DIGIT1:    EQU   $76          ;Tone duration ten seconds
MSDIGIT:   EQU   $77          ;Tone timer counters
LSDIGIT:   EQU   $78          ;
MULTEMP:   EQU   $7A          ;

;          CONSTANTS

RADDL:     EQU   $FC          ;End of RAM Lo byte
RADDH:     EQU   $7F          ;End of RAM Hi byte

;-------------------------------------------------------------
           .ORG $1FF4
;-------------------------------------------------------------
           .DW   INIT         ;SPI
           .DW   INIT         ;SCI
           .DW   TISR         ;Timer
           .DW   INIT         ;IRQ
           .DW   INIT         ;SWI
           .DW   INIT         ;Reset
```

```
;-------------------------------------------------------
          .ORG $1000
;-------------------------------------------------------
          CLR  RAMADDL          ;
          CLR  RAMADDH          ;
          CLR  X                ;
ZC:       JSR  RRAM             ;
          STA  $80,X            ;
          INC  RAMADDL          ;
          INC  X                ;
          CPX  £64              ;
          BNE  ZC               ;
CVB:      BRA  CVB              ;
;-------------------------------------------------------



          .ORG $100
;-------------------------------------------------------

INIT:     SEI                   ;Disable MCU interupts
          RSP                   ;Reset stack pointer
          LDA  £$FF             ;
          STA  DDRA             ;Set up I/O ports
          STA  DDRB             ;Makes PA & PB  outputs

          CLR  PA               ;PA = 0
          CLR  PB               ;PB = 0
          CLR  PRC              ;PC = 0

          LDA  £$F0             ;Port C, Bits 4567 outputs
          STA  DDRC             ;Bits 3210 inputs

;-------------------------------------------------------
;*****    Initilize control bus and MCU ram     ****
;-------------------------------------------------------

          BSET 7,PB             ;CS/EO for battery ram   HI
          BCLR 4,PRC            ;Address Latch Enable    LO
          BSET 5,PRC            ;Ram & LCD  R/W          HI
          BCLR 6,PRC            ;LCD strobe line         LO
          BSET 7,PRC            ;Sine strobe line   LO

          CLR  FLAG             ;Flag=0

;-------------------------------------------------------
;*****    Initilize SPI  &  SCI                 ****
;-------------------------------------------------------

          LDA  £$30             ;Baud rate = 9600/4MHz
          STA  BRR              ;          = 4800/2MHz
          CLR  SCCR1            ;
          LDA  £$0C             ;Enable receiver
          STA  SCCR2            ;

          LDA  £$50             ;
          STA  SPCR             ;
```

```
;-------------------------------------------------------
;****     Initilize RAM addresses & sequence counter ****
;-------------------------------------------------------
          LDA   £$7F            ;
          STA   RAMADDH         ;
          LDA   £$FF            ;
          STA   RAMADDL         ;Point to top of RAM $7FFF
          JSR   RRAM            ;Get contents
          STA   RAMPNTH         ;
          DEC   RAMADDL         ;Next byte $7FFE
          JSR   RRAM            ;
          STA   RAMPNTL         ;
          DEC   RAMADDL         ;Next byte $7FFD
          JSR   RRAM            ;
          STA   SEQCNT          ;Restore sequence number
                                ;counter from ram.

;-------------------------------------------------------
;****     INITIALIZE THE LCD                        ****
;****     LCD ADDRESS  A0=0 & A0=1                  ****
;-------------------------------------------------------



LCDINIT:  JSR   TIMEINIT        ;Start clock, 1ms ticks
          JSR   FCNSET          ;Function set
          JSR   FCNSET          ;Function set
          JSR   FCNSET          ;Function set

          JSR   BNKOFF          ;Display on cursor off

          LDA   £$06            ;Entry mode, INC address
          JSR   WCTRL           ;Write LCD control reg
          JSR   STCLK           ;Start clock
          LDX   £01             ;Delay = 1mS
          JSR   DELAYM          ;Wait 1mS

TOPA:     LDA   £200            ;Setup 200 loops of logo sign
          STA   TIMOUT          ;before beeping every 5 loops
TOPB:     JSR   BEEP3           ;Sound 3 beeps
;-------------------------------------------------------
;*****    FLASH  <<SYNCRO - SPORT>>  MESSAGE        *****
;*****    Wait for Menu select keypress          *****
;-------------------------------------------------------

LOGO:     JSR   DPYCLR          ;Clear LCD Home cursor
          JSR   SYNCRO          ;Display SYNCRO - TECH
          JSR   STCLK           ;Start Millisecond timer
          LDX   £20             ;2 seconds on
FLASH:    BRCLR 7,PD,MSELC      ;Check for keypress
          CPX   HSEC            ;Are they equal
          BNE   FLASH           ;No then check again
          CLR   TCR             ;Stop clock
DPYMENU:  JSR   DPYCLR          ;Clear display and home cursor
          JSR   MENU            ;Display menu
```

```
            JSR   STCLK          ;Start Millisecond timer
            LDX   £30            ;3 seconds on
FLSH:       BRCLR 7,PD,MSELC     ;Check for keypress
            CPX   HSEC           ;Are they equal
            BNE   FLSH           ;No then check again
            CLR   TCR            ;Stop clock
            DEC   TIMOUT         ;Loop counter
            BNE   LOGO           ;Keep checking for a keypress
            LDA   £05            ;Reset loop counter
            STA   TIMOUT         ;for 5 counts
            BRA   TOPB           ;Keep checking for a keypress

;-------------------------------------------------------
;*****    Menue select routine  *****
;-------------------------------------------------------
MSELC:      CLR   TCR            ;Stop clock
            LDA   PRC            ;Read PORTC
            AND   £$0F           ;Mask off top 4 bits
            CMP   £$06           ;If key other than 6,7,E,F
            BEQ   PLYONY         ;selected warning beep sounded
            CMP   £$07           ;program reverts to logo/menu.
            BEQ   RSTMEM         ;Key=$06, play notes no store.
            CMP   £$0E           ;Bit 0,flag=0
            BEQ   PLYSTR         ;Key=$07, Reset memory to $0000
            CMP   £$0F           ;Key=$0E, Play and store notes.
            BEQ   PLYMEM         ;Bit 0,flag=1
                                 ;Key=$0F, play notes in memory.
DPYMNU:     JSR   PEEP           ;Setup beep for 1 beep
            JSR   KEYRL          ;Wait till key released
            BRA   DPYMENU        ;Show menu message again
                                 ;Keep scaning keys



;-------------------------------------------------------
RSTMEM:     JMP   RSTRAM         ;Reset memory to $0000
PLYMEM:     JMP   PLAYNUM        ;Play what's in memory
TNEND:      JMP   TONEND         ;End tone sequence
SETFLG:     JMP   SETC           ;
CLRFLG      JMP   CLRC           ;
;-------------------------------------------------------
PLYONY:     BSET  0,FLAG         ;Play & store flag = 1
            JSR   PLYNSTR        ;Set for play only
            JSR   BNKON          ;
            BRA   UP             ;
;-------------------------------------------------------
PLYSTR:     BCLR  0,FLAG         ;Play & store flag = 0

            LDA   SEQCNT         ;If 1st sequence put end of
            CMP   £$30           ;sequnce marker in 1st ram loc
            BNE   NOTFST         ;Then 2nd loc is sequence number

            LDA   £$FF           ;$FF is placed in ram 1st byte
            STA   SCALE          ;to signify that the next byte
            JSR   STOREIT        ;is the sequence number

NOTFST:     INC   SEQCNT         ;Add 1 to sequence number and
            JSR   SEQSAVE        ;store it in RAM $7FFD
```

```
            LDA   SEQCNT          ;Get the sequence number
            STA   SCALE           ;and store it
            JSR   STOREIT         ;in next free RAM byte

            JSR   BNKOFF          ;
            JSR   DPYSEQ          ;Display 'SEQ No: '
            LDA   SEQCNT          ;Sequence number address
            JSR   WLCD            ;Display sequence number
            LDX   £25             ;for 2.5 seconds

            JSR   DELAYH          ;
            JSR   BNKON           ;
;-----------------------------------------------------------
;****      Tone selection program starts here          ****
;-----------------------------------------------------------
            LDA   £$03            ;Counter for positioning data in
            STA   DATACNT         ;LCD.  Initial value =$03
            CLR   DPLYPNT         ;Clear rowb position counter
            JSR   NOTES           ;Notes message
UP:         JSR   KEYRL           ;Wait till key released
            JSR   NTKEY           ;Get note from keys
            STA   TEMP            ;Save it
            CMP   £$0E            ;Set flag for upper tones
            BEQ   SETFLG          ;
            CMP   £$06            ;Clear flag for lower tones
            BEQ   CLRFLG          ;
            CMP   £$0F            ;Exit collect notes routine
            BEQ   TNEND           ;if key =$0F
;-----------------------------------------------------------
TONES:      LDX   £72             ;Point to last line in table.
NXTKEY:     CMP   KEYTBL,X        ;Compare to 1st colum in table.
            BEQ   FOUND           ;Got a match, go found
            DEC   X               ;No then point to
            DEC   X               ;next line in table.
            DEC   X               ;
            DEC   X               ;
            DEC   X               ;



            DEC   X               ;
            BRA   NXTKEY          ;No match, then try again.
;-----------------------------------------------------------
FOUND:      LDA   KEYTBL+1,X      ;Get note value (2nd column)
            JSR   WLCD            ;Display in next LCD position
            JSR   TSTSHP          ;test for a sharp, note in TEMP
            BRSET 1,FLAG,HIGHC    ;High or low note
;-----------------------------------------------------------
            LDA   KEYTBL+2,X      ;Note value (3rd & 4th column)
            STA   MSBY            ;
            LDA   KEYTBL+3,X      ;
            STA   LSBY            ;
            BRA   SCLE            ;
```

```
;-------------------------------------------------------
HIGHC:    LDA  KEYTBL+4,X     ;Note value (5th & 6th column)
          STA  MSBY           ;
          LDA  KEYTBL+5,X     ;
          STA  LSBY           ;
;-------------------------------------------------------
SCLE:     BRSET 0,FLAG,JPONY   ;If flag set play only
          LDA  MSBY           ;
          STA  SCALE          ;
          JSR  STOREIT        ;Store tone frequency
          LDA  LSBY           ;
          STA  SCALE          ;
          JSR  STOREIT        ;Store tone frequency
;-------------------------------------------------------
;         The following code gets the period of the tone
;         (3 bytes) displays and stores it ready for ascon
;-------------------------------------------------------
          JSR  TSTKEY         ;Get & test 1st key write to LCD
          LDA  TEMPA          ;
          STA  DIGIT2         ;Save 1st digit ready for ascon

          LDA  £$2E           ;Decimal point
          JSR  WLCD           ;Write to LCD

          JSR  TSTKEY         ;Get & test 2nd key write to LCD
          LDA  TEMPA          ;
          STA  DIGIT3         ;Save 2nd digit ready for ascon

          JSR  TSTKEY         ;Get & test 3rd key write to LCD
          LDA  TEMPA          ;
          STA  DIGIT4         ;Save 3rd digit ready for ascon

          LDA  £$30           ;Put zero on end of digits
          JSR  WLCD           ;in LCD
;-------------------------------------------------------
;         The following code takes the 5 BCD digits
;         stored in DIGIT1-5 and converts them to binary
;         The result is stored in MSDIGIT & LSDIGIT.
;         Digits 1 & 5 are always zero.
;-------------------------------------------------------

          CLR  DIGIT1         ;Digits 1 & 5 always zero
          CLR  DIGIT5         ;eg. tone on =01.230 secs

          CLR  MSDIGIT        ;Clear upper byte
          LDA  DIGIT1         ;Get most significant digit
          STA  LSDIGIT        ;Store in lower byte

          LDX  £$04           ;Set index for 4 digits



NXTDIG:   LDA  DIGIT5-1,X     ;Get next digit
          JSR  MULTEN         ;
          DEC  X              ;
          BNE  NXTDIG         ;
```

```
                LDA   MSDIGIT          ;
                STA   SCALE            ;
                JSR   STOREIT          ;
                LDA   LSDIGIT          ;
                STA   SCALE            ;
                JSR   STOREIT          ;
                BRA   LCDATA           ;

JPONY:          JMP   PONY             ;Relitive jump to large

MULTEN:         STA   MULTEMP          ;
                LDA   MSDIGIT          ;
                STA   TEMPM            ;
                LDA   LSDIGIT          ;
                STA   TEMPL            ;
                ASL   LSDIGIT          ;
                ROL   MSDIGIT          ;
                ASL   LSDIGIT          ;
                ROL   MSDIGIT          ;
                LDA   TEMPL            ;
                ADC   LSDIGIT          ;
                STA   LSDIGIT          ;
                LDA   TEMPM            ;
                ADC   MSDIGIT          ;
                STA   MSDIGIT          ;
                ASL   LSDIGIT          ;
                ROL   MSDIGIT          ;
                LDA   MULTEMP          ;
                ADC   LSDIGIT          ;
                STA   LSDIGIT          ;
                CLR   A                ;
                ADC   MSDIGIT          ;
                STA   MSDIGIT          ;
                RTS                    ;
;-------------------------------------------------------
;     The following code positions the notes and times
;     in the LCD eg. (A  1.110    B  2.220)  X 2 lines
;-------------------------------------------------------
LCDATA:         DEC   DATACNT          ;Point to next data group in LCD
                LDA   £$03             ;If =03 then put 4 spaces in LCD
                CMP   DATACNT          ;
                BEQ   SPFOUR           ;
                DEC   A                ;If =02 then go to line 2 in LCD
                CMP   DATACNT          ;
                BEQ   SETROWB          ;
                DEC   A                ;
                CMP   DATACNT          ;If =01 then put $ spaces in LCD
                BEQ   SPFOUR           ;
                JSR   ROWA             ;Ifnot 3,2, or 1 then must be 00
                LDA   £$04             ;Initialise data counter to $04
                STA   DATACNT          ;

FULTST:         BRSET 2,FLAG,RFUL      ;Ram full if 2,flag is set
PONY:           JSR   TONE             ;Play note no store
                JMP   UP               ;Next note
;-------------------------------------------------------
RFUL:           JMP   DPYMNU           ;Return to the menu
```

```
;-------------------------------------------------------
TONEND:   BRSET 0,FLAG,TEND   ;Return to menu if tone only
          LDA  £$FF           ;Put $FF in next free byte to
          STA  SCALE          ;
          JSR  STOREIT        ;indicate end of a tone sequence
TEND:     JMP  DPYMNU         ;Return to the menu
;-------------------------------------------------------
SETC:     BSET 1,FLAG         ;Set high tones flag
          JMP  UP             ;Return to key checking routine
CLRC:     BCLR 1,FLAG         ;Clear high tones flag
          JMP  UP             ;Return to key checking routine
;-------------------------------------------------------
SETROWB:  JSR  ROWB           ;Set lcd to 2nd line
          BRA  FULTST         ;Return to main routine
;-------------------------------------------------------
SPFOUR:   LDX  £$04           ;Output 4 spaces to LCD
MORES:    LDA  £$20           ;
          JSR  WLCD           ;
          DEC  X              ;
          BNE  MORES          ;
          BRA  FULTST         ;Return to main routine
;-------------------------------------------------------
TSTKEY:   JSR  KEYRL          ;Wait for key release
          JSR  NTKEY          ;
          CMP  £$0C           ;Key > C then no good
          BHI  ERROR          ;
          CMP  £$08           ;Key < 8 then more tests needed
          BLO  TEST           ;
          BRA  OK             ;8 < key < C  then key is ok
TEST:     CMP  £$05           ;
          BLO  OK             ;If key < 5 then key is ok
ERROR:    JSR  PEEP           ;Beep if key is wrong
          BRA  TSTKEY         ;Then test next key

OK:       LDX  £27            ;Key ok then search table
NXTLIN:   CMP  TIMTBL,X       ;for a match
          BEQ  AOK            ;Found one
          DEC  X              ;
          DEC  X              ;
          DEC  X              ;
          BRA  NXTLIN         ;No then keep looking

AOK:      LDA  TIMTBL+2,X     ;Ascii numbers for LCD
          JSR  WLCD           ;

          LDA  TIMTBL+1,X     ;Hex numbers for timer
          STA  TEMPA          ;Save time.
          RTS                 ;
;-------------------------------------------------------
TSTSHP:   LDA  TEMP           ;Get the key value
          CMP  £$00           ;Check for a sharp key
          BEQ  SETSHP         ;
          CMP  £$03           ;
          BEQ  SETSHP         ;
          CMP  £$05           ;
          BEQ  SETSHP         ;
          CMP  £$09           ;
          BEQ  SETSHP         ;
          CMP  £$0C           ;
```

```
            BEQ   SETSHP          ;
            BCLR  3,FLAG          ;No sharps
            BRA   NOSHP           ;


SETSHP:     BSET  3,FLAG          ;Sharp note
            LDA   £$DF            ;Places a £ symbol after
            JSR   WLCD            ;D,F,G,A & C in LCD
            BRCLR 0,FLAG,RETS     ;If set then play & store
            INC   DPLYPNT         ;Point to next vac LCD position
NOSHP:      BRCLR 0,FLAG,RTNS     ;If set then play & store
            INC   DPLYPNT         ;Point to next vac LCD position
            LDA   £20             ;20 positions in 2nd line
            CMP   DPLYPNT         ;At the end yet
            BHI   RETNS           ;No then get next note
            CLR   DPLYPNT         ;Clear rowb position counter
            JSR   ROWB            ;Yes then 2nd row of LCD
RETNS:      RTS                   ;Return

RTNS:       LDA   £$20            ;Output two spaces to LCD
            JSR   WLCD            ;
RETS:       LDA   £$20            ;Output one space to LCD
            JSR   WLCD            ;
            RTS                   ;Return

;*************************************************************
;*          Play note sequence                               *
;*************************************************************

PLAYNUM:    JSR   PLAYMESG        ;Play it message
UPIT:       JSR   KEYRL           ;Wait till key released
            JSR   NTKEY           ;Get sequence number from keys
            CMP   £$05            ;Compare to $05
            BLS   GOOD            ;Branch if less than 6
            CMP   £$06            ;06,07 not valid keys
            BEQ   NOGOOD          ;
            CMP   £$07            ;
            BEQ   NOGOOD          ;
            CMP   £$0A            ;0B,0C,0D,0E,0F
            BLS   GOOD            ;are not valid keys
NOGOOD:     JSR   PEEP            ;Warning tone
            BRA   UPIT            ;Keep looking for a valid key

GOOD:       LDX   £16             ;Point to last line in table.
SEQUM:      CMP   SEQTBL,X        ;Compare to 1st col in table.
            BEQ   FND             ;Got a match, go found
            DEC   X               ;No then point to
            DEC   X               ;next line in table.
            BRA   SEQUM           ;No match, then try again.

FND:        LDA   SEQTBL+1,X      ;Get sequence Num (2nd column)
            STA   TEMP            ;Store it for a moment
            JSR   WLCD            ;Display in next LCD position
            CLR   RAMADDL         ;Start at $0000
            CLR   RAMADDH         ;
            JSR   RRAM            ;Read the 1st location
```

```
            CMP   £$AA              ;If memory has $AA in 1st
            BEQ   MEMCLR            ;location then memory is clear

FINDSEQ:    CMP   £$FF              ;
            BEQ   FOUNDIT           ;Play the sequence
            LDA   RAMADDH           ;
            CMP   £$7C              ;End of usable RAM yet?
            BEQ   PANIC             ;Then end search
FNDSEQ:     BSR   NXTBYTE           ;
            BRA   FINDSEQ           ;Keep looking



NXTBYTE:    INC   RAMADDL           ;Next byte
            BNE   RAMR              ;$FF bytes done yet
            INC   RAMADDH           ;Yes new block then
RAMR:       JSR   RRAM              ;No, read the address
            RTS                     ;

FOUNDIT:    BSR   NXTBYTE           ;
            CMP   TEMP              ;
            BNE   FNDSEQ            ;
            JSR   SHOWIT            ;Display 'FOUND SEQUENCE No.'
            LDA   RAMDATA           ;Get the sequence number
            JSR   WLCD              ;Display the number
;-----------------------------------------------------------
;     Data format in ram is 4 bytes long.  The 1st & 2nd
;     bytes contain the tone frequency. The 3rd & 4th byte
;     has the period in multiples of 10 milliseconds
;-----------------------------------------------------------
            BSET 6,FLAG             ;Set tone timer flag
NXTONE:     BSR   NXTBYTE           ;Get byte from RAM
            CMP   £$FF              ;Is it the end of sequence?
            BEQ   SQUEND            ;Yes then end sequence
            STA   MSBY              ;No then 1st byte of tone
            BSR   NXTBYTE           ;
            STA   LSBY              ;2nd byte of tone
            BSR   NXTBYTE           ;
            STA   MSDIGIT           ;Tone duration HI byte
            INC   MSDIGIT           ;
            BSR   NXTBYTE           ;
            STA   LSDIGIT           ;Tone duration LO byte
            INC   LSDIGIT           ;
            BCLR 7,FLAG             ;Clear time up flag
            JSR   TONE              ;Play tone
            JSR   STCLK             ;Start timer
TONON:      BRCLR 7,FLAG,TONON      :Wait till finished
            CLR   TCR               ;Stop timer
            BRA   NXTONE            ;Next note in sequence

;-----------------------------------------------------------
MEMCLR:     JSR   CLRMEM            ;Display '    MEMORY EMPTY    '
            BRA   SQUEND            ;

PANIC:      JSR   FAULT             ;Display ' 32K BYTES SEARCHED '
```

```
SQUEND:   CLR  MSBY           ;
          CLR  LSBY           ;
          JSR  TONE           ;Inhibit tone generator
          BCLR 6,FLAG         ;Set for normal timer counters
          LDX  £10            ;Wait 1.0 sec before sounding
          JSR  DELAYH         ;3 beeps and returning to menu.
          JMP  TOPA           ;Show menu message again
                              ;Finished playing sequence.
;-------------------------------------------------------
;****     Get Key program starts here            ****
;-------------------------------------------------------

NTKEY:    BRSET 7,PD,NTKEY    ;Wait for key press
          LDA  PRC            ;Get key
          AND  £$0F           ;Mask off top bits
          RTS                 ;Return, $00-$0F in ACC

GETKEY:   BRSET 7,PD,GETKEY   ;Wait for key press



          LDA  PRC            ;Get key
          AND  £$0F           ;Mask off top bits
          ADD  £$30           ;Convert hex to ASCII
          JSR  WLCD           ;Display note in LCD
          JSR  KEYRL          ;Wait for key release
          RTS                 ;Return

;*******************************************************
;*        TONE GENERATION SUBROUTINES                  *
;*******************************************************

PEEP:     LDX  £18            ;Point to 1st tone in table
          LDA  £20            ;No of tones to play (1)
          STA  BEEPS          ;(value of X+No. of tones)
          BRA  BEEP           ;

BEEP3:    LDX  £12            ;Point to 1st tone in table
          LDA  £18            ;No of tones to play (3)
          STA  BEEPS          ;(value of X+No. of tones)

BEEP:     LDA  TONETBL,X ;Get MSBY of tone frequency from
          STA  MSBY           ;table and Save in MSBY
          INC  X              ;Point to LSBY
          LDA  TONETBL,X ;Get LSBY of tone frequency from
          STA  LSBY           ;table and Save in MSBY
          JSR  TONE           ;Play the tone
          STX  TEMPX          ;Save X
          JSR  STCLK          ;Start the timer
          LDX  £02            ;for a 300 mS period
LOOP:     CPX  HSEC           ;
          BNE  LOOP           ;Time up? no check again
          CLR  TCR            ;Stop the timer
          LDX  TEMPX          ;Yes then restore X
          INC  X              ;Point to next tone
          CPX  BEEPS          ;Check if more tones
          BNE  BEEP           ;No more tones?
```

```
            CLR   MSBY              ;Clear tone stores
            CLR   LSBY              ;Then stop the tones
            BRA   SENDT             ;

;-------------------------------------------------------
;           TONE  Swaps the order of the bytes
;           SENDT Sends the tones to tone generator
;-------------------------------------------------------

TONE:       LDA   MSBY              ;Save MSBY
            STA   TEMPA             ;
            LDA   LSBY              ;Get the LSBY
            STA   NUMBER            ;
            JSR   SWAP              ;Mirror bits
            LDA   TEMP              ;
            STA   MSBY              ;Save mirrored bits in MSBY
            LDA   TEMPA             ;
            STA   NUMBER            ;Retrive the MSBY
            JSR   SWAP              ;Mirror bits
            LDA   TEMP              ;
            STA   LSBY              ;Save mirrored bits in LSBY
;-------------------------------------------------------
SENDT:      LDA   MSBY              ;Load LSB to the SPI data
            STA   SPDR              ;register and initiate transfer
HERE:       BRCLR 7,SPSR,HERE       ;Wait till finished
            LDA   LSBY              ;Load MSB to the SPI data



            STA   SPDR              ;register and initiate transfer
ERE:        BRCLR 7,SPSR,ERE        ;Wait till finished
            BSET  7,PRC             ;Strobe in data
            BCLR  7,PRC             ;
            RTS                     ;
;-------------------------------------------------------
;     This routine takes the binary value in NUMBER and
;     produces the mirror immage of the bits. The result of
;     this bit manipulation is left in TEMP.
;-------------------------------------------------------
SWAP:       LDA   £$08              ;Loop counter
LOOP1:      ROR   NUMBER            ;Rotate bit into carry bit
            ROL   TEMP              ;Rotate carry into next bit
            DEC   A                 ;8 bits yet
            BNE   LOOP1             ;
            RTS                     ;

;*******************************************************
;*                                                     *
;*         BATTERY RAM SUBROUTINES                     *
;*                                                     *
;*   RAM address range $0000 - $7FFF                   *
;*   RAM address $7FFE & $7FFF contains pointer to 1st *
;*   vacant RAM address.                               *
;*   RAMDATA contains the data to be writen to and read *
;*   from the RAM.  The 1st free location holds $AA    *
;*                                                     *
;*******************************************************
```

```
SETADDR:  LDA   £$FF          ;Port A
          STA   DDRA          ;Outputs
          LDA   RAMADDL       ;Lo address
          STA   PA            ;
          JSR   ALE           ;Latched
          LDA   RAMADDH       ;Hi address
          ORA   £$80          ;Make sure PB7 always set
          STA   PB            ;
          RTS                 ;
;-------------------------------------------------------
WRAM:     JSR   SETADDR       ;Set up address for the data
          LDA   RAMDATA       ;Fetch data
          STA   PA            ;Output to data bus
          BCLR  5,PRC         ;R/W = 0
          BCLR  7,PB          ;CS/EO=0
          BSET  7,PB          ;CS/EO=1
          BSET  5,PRC         ;R/W = 1
          RTS                 ;Return
;-------------------------------------------------------
RRAM:     JSR   SETADDR       ;Read cycle No 3
          CLR   DDRA          ;Make PORTA input
          BCLR  7,PB          ;OE=0 enable output
          LDA   PA            ;Read RAM data
          STA   RAMDATA       ;Save data
          BSET  7,PB          ;OE=1
          RTS                 ;Return

;*******************************************************
;*   Routine to store data in battery backed ram       *
;*   At the end of this subroutine addresses $7FFF & E *
;*   contain the address of the next free byte in ram. *
;*   The data in this byte is $AA.                     *
;*******************************************************


STOREIT:  LDA   £$7F          ;
          STA   RAMADDH       ;
          LDA   £$FF          ;$7FFF = Address of ram pointer
          STA   RAMADDL       ;Hi byte
          JSR   RRAM          ;Get the contents of $7FFF for
          STA   RAMPNTH       ;RAM pointer Hi byte.

          DEC   RAMADDL       ;$7FFE = Lo byte
          JSR   RRAM          ;Read RAM address $7FFE for
          STA   RAMPNTL       ;RAM pointer Lo byte.

;RAMPNT H&L now contain the address of next free ram byte
;This address has to be put into RAMADDR H & L

          STA   RAMADDL       ;Put Lo byte in RAMADDL
          LDA   RAMPNTH       ;
          STA   RAMADDH       ;Put Hi byte in RAMADDH
```

```
            LDA   SCALE           ;Get tone frequency
            STA   RAMDATA         ;Put it where WRAM can get it
            JSR   WRAM            ;Store it in RAM

;**  The following code increments ram address counter  *
;** and stores it back to $7FFF & $7FFE, next free byte *

            INC   RAMPNTL         ;Point to next ram address
            BNE   AWAY            ;= 00 yet? No, then away
            INC   RAMPNTH         ;Yes, then point to next block
            BMI   RAMFULL         ;The battery ram is full

AWAY:       LDA   £$7F            ;Set RAM address to $7FFF
            STA   RAMADDH         ;
            LDA   £$FF            ;
            STA   RAMADDL         ;
            LDA   RAMPNTH         ;Get next RAM address Hi byte
            STA   RAMDATA         ;Put it where WRAM can get it
            JSR   WRAM            ;Store it in RAM ($7FFF)

            DEC   RAMADDL         ;Set RAM addres to $7FFE
            LDA   RAMPNTL         ;Get next RAM address Lo byte
            STA   RAMDATA         ;Put it where WRAM can get it
            JSR   WRAM            ;Store it in RAM ($7FFE)
            RTS                   ;Return
;-------------------------------------------------------
RAMFULL     JSR   RAMFUL          ;
            JSR   BEEP3           ;3 beeps
            LDX   £$15            ;
            JSR   DELAYM          ;1.5 SEC delay
            JSR   BEEP            ;3 beeps
            BSET  2,FLAG          ;Set RAM full flag
            RTS                   ;

;*******************************************************
;*   The following code writes $FF then $00 then $55   *
;*   then $AA in turn to all 32K bytes of RAM.         *
;*   The sequence counter is reset to zero ($7FFD).    *
;*   $AA is then writen to the 1st byte of RAM.        *
;*******************************************************

RSTRAM:     JSR   TESTMSGA        ;RAM message
            CLR   RAMADDH         ;Point to bottom of RAM



            CLR   RAMADDL         ;
BLOCK:      JSR   TESTING         ;Test RAM
            INC   RAMADDL         ;Next byte
            BNE   BLOCK           ;$00 yet? No then do some more
            INC   RAMADDH         ;Yes then next block
            BPL   BLOCK           ;$7FFF bytes done yet

            JSR   TESTMSGB        ;Finished message
            LDX   £20             ;
            JSR   DELAYH          ;Display for 2 secs
```

```
                LDA   £$7F              ;
                STA   RAMADDH           ;
                LDA   £$FF              ;Point to top of battery
                STA   RAMADDL           ;RAM
                CLR   RAMDATA           ;
                JSR   WRAM              ;Put $00 into $7FFF
                DEC   RAMADDL           ;
                JSR   WRAM              ;Put $00 into $7FFE
                DEC   RAMADDL           ;
                LDA   £$30              ;
                STA   SEQCNT            ;Initialise sequence counter
                STA   RAMDATA           ;and $7FFD to 0
                JSR   WRAM              ;
                CLR   RAMADDH           ;Set RAM address counters to
                CLR   RAMADDL           ;Zero
                CLR   RAMPNTH           ;
                CLR   RAMPNTL           ;
                JMP   TOPA              ;Sound 3 beeps return to menue.
;-------------------------------------------------------
TESTING:        LDA   £$FF              ;
                STA   RAMDATA           ;
                JSR   WRAM              ;Write $FF to RAM
                CLR   RAMDATA           ;Destroy contents of RAMDATA
                JSR   RRAM              ;Read RAM
                LDA   £$FF              ;
                CMP   RAMDATA           ;Does it =$FF
                BEQ   A                 ;Yes then keep going Z=1
                JMP   FAULTY            ;No then fault       Z=0
A:              CLR   RAMDATA           ;
                JSR   WRAM              ;Write $00 to RAM
                LDA   £$FF              ;
                STA   RAMDATA           ;Destroy contents of RAMDATA
                JSR   RRAM              ;Read RAM
                CLR   A                 ;
                CMP   RAMDATA           ;Does it =$00
                BEQ   B                 ;Yes then keep going Z=1
                JMP   FAULTY            ;No then fault       Z=0
B:              LDA   £$55              ;
                STA   RAMDATA           ;
                JSR   WRAM              ;Write $55 to RAM
                CLR   RAMDATA           ;Destroy contents of RAMDATA
                JSR   RRAM              ;Read RAM
                LDA   £$55              ;Does it =$55
                CMP   RAMDATA           ;Yes then keep going Z=1
                BEQ   C                 ;No then fault       Z=0
                JMP   FAULTY            ;
C:              LDA   £$AA              ;
                STA   RAMDATA           ;
                JSR   WRAM              ;Write $AA to RAM
                CLR   RAMDATA           ;Destroy contents of RAMDATA
                JSR   RRAM              ;Read RAM
```

```
            LDA   £$AA              ;
            CMP   RAMDATA           ;Does it =$AA
            BEQ   D                 ;Yes then keep going Z=1
            JMP   FAULTY            ;No then fault       Z=0
D:          RTS                     ;Last test leaves RAM
                                    ;location = $AA
FAULTY:     JSR   WRNMSG            ;Sound warning beep
            RTS                     ;
;-------------------------------------------------------
SEQSAVE:    LDA   £$7F              ;Sequence number address
            STA   RAMADDH           ;in RAM= $7FFD
            LDA   £$FD              ;
            STA   RAMADDL           ;
            LDA   SEQCNT            ;Get sequence count
            STA   RAMDATA           ;Put it where WRAM can get it
            JSR   WRAM              ;Write to RAM ($7FFF)
            RTS                     ;Return
;-------------------------------------------------------
SEQGET:     LDA   £$7F              ;Sequence number address
            STA   RAMADDH           ;in RAM= $7FFD
            LDA   £$FD              ;
            STA   RAMADDL           ;
            JSR   RRAM              ;Get number in RAM
            STA   SEQCNT            ;Put it in counter
            RTS                     ;Return

;*********************************************************
;*                                                       *
;*        LIQUID CRYSTAL DISPLAY SUBROUTINES             *
;*   LCDIR   Sets address                AO=0 RS=0       *
;*   LCDDR   Sets address                AO=1 RS=1       *
;*   WCTRL   Writes to control register      R/W=0       *
;*   RCTRL   Reads control register      R/W=1           *
;*   WLCD    Writes to data register     R/W=0           *
;*   PORTC   Bit6 Provides strobe pulse for LCD          *
;*   LCBUSY  DB7=1 Then LCD Busy. Returns when DB7=0     *
;*   NOTE:   All LCD subroutines should leave control    *
;*           lines in original state.                    *
;*********************************************************

ROWA:       LDA   £$80              ;Set cursor to 1st row
            BRA   WCTRL             ;
ROWB:       LDA   £$C0              ;Set cursor to 2nd row
WCTRL:      JSR   LCDIR             ;Set LCD IR address AO=0/RS=0
            STA   PA                ;Write control word to LCD
            BCLR  5,PRC             ;R/W = 0
            BSET  6,PRC             ;E = 1  Strobe E line on LCD
            BCLR  6,PRC             ;E = 0
            BSET  5,PRC             ;R/W = 1
            RTS                     ;Return
;-------------------------------------------------------
RCTRL:      JSR   LCDIR             ;Set LCD IR address   (RS =0)
            CLR   PA                ;
            STA   DDRA              ;PORTA=input
            BSET  6,PRC             ;Strobe E line on LCD (R/W=1)
```

```
            LDA  PA              ;Read
            BCLR 6,PRC           ;E = 0
            RTS                  ;Returns with LCD data in ACC

;-------------------------------------------------------
WLCD:       JSR  LCDDR           ;Set LCD IR address (RS =1)
            STA  PA              ;Write data word to LCD



            BCLR 5,PRC           ;R/W = 0
            BSET 6,PRC           ;Strobe E line on LCD
            BCLR 6,PRC           ;E = 0
            BSET 5,PRC           ;R/W = 1
            RTS                  ;Return

;-------------------------------------------------------
LCDIR:      JSR  SAVE            ;Save A&X
            LDA  £$FF            ;LCD Instruction Register
            STA  DDRA            ;Set Port A output
            CLR  PA              ;PORTA = 00     RS=0
            JSR  ALE             ;Latch address to LO Bus
            JSR  RESTORE         ;Restore A&X
            RTS                  ;Return

;-------------------------------------------------------
LCDDR:      JSR  SAVE            ;Save A&X
            LDA  £$FF            ;LCD Data Register
            STA  DDRA            ;Set port A output
            LDA  £$01            ;
            STA  PA              ;PORTA = 01     RS=1
            JSR  ALE             ;Latch address to LO Bus
            JSR  RESTORE         ;Restore A&X
            RTS                  ;Return

;-------------------------------------------------------
DPYCLR:     LDA  £$01            ;Clear display.
            JSR  WCTRL           ;Write LCD Control Reg.
            LDX  £10             ;
            JSR  DELAYM          ;Wait till done.
HOME:       LDA  £$02            ;Return cursor to home position
            JSR  WCTRL           ;Write LCD Control Reg.
            LDX  £10             ;
            JSR  DELAYM          ;Wait till done.
            RTS                  ;Return.

;-------------------------------------------------------
LCBSY:      JSR  RCTRL           ;Read instruction register
            BRSET 7,PA,LCBSY     ;TEST DB7=1 FOR BUSY
            RTS                  ;Return
;-------------------------------------------------------
FCNSET:     LDA  £$38            ;Function set
            JSR  WCTRL           ;Write LCD Control REG
            LDX  £10             ;Delay = 10mS
            JSR  DELAYM          ;Wait 10mS
            RTS
```

```
;-------------------------------------------------------
BNKON:    LDA   £$OD            ;Display on/off control
          JSR   WCTRL           ;Display on cursor off
          LDX   £01
          JSR   DELAYM          ;Wait 1mS
          RTS                   ;
;-------------------------------------------------------
BNKOFF:   LDA   £$OC            ;Display on/off control
          JSR   WCTRL           ;Display on cursor off
          LDX   £01
          JSR   DELAYM          ;Wait 1mS
          RTS                   ;

;*******************************************************
;*        MISCELANEOUS SUBROUTINES                     *
;*        ADDRESS LATCH ENABLE,                        *



;*        SAVE A&X, RESTORE A&X,                       *
;*        KEYRL, DELAY                                 *
;*******************************************************

ALE:      BSET 4,PRC            ;ALE = 1
          BCLR 4,PRC            ;ALE = 0
          RTS                   ;Return
;-------------------------------------------------------
SAVE:     STA   TEMPA           ;Save A
          STX   TEMPX           ;Save X
          RTS                   ;Return
;-------------------------------------------------------
RESTORE:  LDA   TEMPA           ;Restore A
          LDX   TEMPX           ;Restore X
          RTS                   ;Return
;-------------------------------------------------------
;DELAYM Uses X and MSEC to give a varrible length delay
;       in 0.001 SEC increments.
;
;DELAY Uses X and HSEC to give a varrible length delay
;       in 0.1 SEC increments.
;-------------------------------------------------------
DELAYM:   JSR   TIMEINIT        ;Start clock
LOOP2:    CPX   MSEC            ;Compare X with LOW counter
          BNE   LOOP2           ;Loop till equal
          BRA   RTN             ;Stop clock and return
;-------------------------------------------------------
DELAYH:   JSR   STCLK           ;Start clock
LOOP3:    CPX   HSEC            ;Compare X with HIGH counter
          BNE   LOOP3           ;Loop till equal
RTN:      CLR   TCR             ;Stop clock
          RTS                   ;Return
;-------------------------------------------------------
KEYRL:    BRCLR 7,PD,KEYRL      ;Wait till key released
          RTS                   ;Return
```

```
;*****************************************************
;*        LCD Messages                               *
;*****************************************************

SYNCRO:   CLR  X              ;X=0
REPTA:    LDA  SYNC,X         ;Get character
          JSR  WLCD           ;Display it
          INC  X              ;
          CPX  £20            ;20 chrs yet
          BCS  REPTA          ;No keep going
          JSR  ROWB           ;2nd line
          CLR  X              ;X=0
RPTA:     LDA  MENUE,X        ;Get character
          JSR  WLCD           ;Display it
          INC  X              ;
          CPX  £20            ;20 chrs yet
          BCS  RPTA           ;No keep going
          RTS                 ;Return
;-----------------------------------------------------

MENU:     CLR  X              ;X=0
REPTB:    LDA  SELECT1,X      ;Get character
          JSR  WLCD           ;Display it
          INC  X              ;
          CPX  £20            ;20 chrs yet
          BCS  REPTB          ;No keep going



          JSR  ROWB           ;2nd line
          CLR  X              ;X=0
RPTB:     LDA  SELECT2,X      ;Get character
          JSR  WLCD           ;Display it
          INC  X              ;
          CPX  £20            ;20 chrs yet
          BCS  RPTB           ;No keep going
          RTS                 ;Return
;-----------------------------------------------------

NOTES:    JSR DPYCLR          ;Clear LCD
          CLR  X              ;X=0
REPTC:    LDA  NOTE,X         ;Get character
          JSR  WLCD           ;Display it
          INC  X              ;
          CPX  £12            ;12 chrs yet
          BCS  REPTC          ;No keep going
          RTS                 ;Return. LCD cusor left at next
;-----------------------------------------------------
                              ;position.
PLAYMESG: JSR DPYCLR          ;Clear LCD
          CLR  X              ;X=0
REPTD:    LDA  PLYMSG,X       ;Get character
          JSR  WLCD           ;Display it
          INC  X              ;
          CPX  £15            ;15 chrs yet
          BCS  REPTD          ;No keep going
          RTS                 ;Return. LCD cusor left at next
```

```
;-------------------------------------------------------
                                    ;position.
CLRMEM:   JSR  ROWB                 ;Clear LCD
          CLR  X                    ;X=0
REPTE:    LDA  CLRMSG,X             ;Get character
          JSR  WLCD                 ;Display it
          INC  X                    ;
          CPX  £20                  ;20 chrs yet
          BCS  REPTE                ;No keep going
          RTS                       ;Return.
;-------------------------------------------------------

ZEROMSG:  CLR  X                    ;X=0
REPTF:    LDA  ZERO,X               ;Get character
          JSR  WLCD                 ;Display it
          INC  X                    ;
          CPX  £20                  ;20 chrs yet
          BCS  REPTF                ;No keep going
          JSR  ROWB                 ;2nd line
          CLR  X                    ;X=0
RPTG:     LDA  RESET,X              ;Get character
          JSR  WLCD                 ;Display it
          INC  X                    ;
          CPX  £20                  ;20 chrs yet
          BCS  RPTG                 ;No keep going
          RTS                       ;Return
;-------------------------------------------------------

FINIS:    JSR DPYCLR                ;Clear LCD
          CLR  X                    ;X=0
REPTH:    LDA  FIN,X                ;Get character
          JSR  WLCD                 ;Display it
          INC  X                    ;
          CPX  £20                  ;20 chrs yet

          BCS  REPTH                ;No keep going
          RTS                       ;Return. LCD cusor left at next
                                    ;position.
;-------------------------------------------------------
FAULT:    JSR  ROWB                 ;Clear LCD
          CLR  X                    ;X=0
REPTI:    LDA  FALTY,X              ;Get character
          JSR  WLCD                 ;Display it
          INC  X                    ;
          CPX  £20                  ;20 chrs yet
          BCS  REPTI                ;No keep going
          RTS                       ;Return.
;-------------------------------------------------------

SHOWIT:   JSR  ROWB                 ;
          CLR  X                    ;X=0
REPTJ:    LDA  FNDIT,X              ;Get character
          JSR  WLCD                 ;Display it
          INC  X                    ;
          CPX  £18                  ;18 chrs yet
          BCS  REPTJ                ;No keep going
          RTS                       ;Return. LCD cusor left at next
                                    ;position.
```

```
;-----------------------------------------
PLYNSTR:  JSR  DPYCLR          ;Clear LCD
          CLR  X               ;X=0
REPTK:    LDA  ONLYPLY,X ;Get character
          JSR  WLCD            ;Display it
          INC  X               ;
          CPX  £20             ;20 chrs yet
          BCS  REPTK           ;No keep going
          JSR  ROWB            ;2nd line of display
          RTS                  ;Return.
;-----------------------------------------------------
TESTMSGA: JSR DPYCLR           ;Clear LCD
          CLR  X               ;X=0
REPTU:    LDA  MESGA,X         ;Get character
          JSR  WLCD            ;Display it
          INC  X               ;
          CPX  £20             ;20 chrs yet
          BCS  REPTU           ;No keep going
          RTS                  ;Return. LCD cusor left at next
                               ;position.
;-----------------------------------------------------
TESTMSGB: JSR ROWB             ;2nd line of display
          CLR  X               ;X=0
REPTV:    LDA  MESGB,X         ;Get character
          JSR  WLCD            ;Display it
          INC  X               ;
          CPX  £20             ;20 chrs yet
          BCS  REPTV           ;No keep going
          RTS                  ;Return. LCD cusor left at next
                               ;position.
;-----------------------------------------------------
DPYSEQ:   JSR DPYCLR           ;Clear LCD
          CLR  X               ;X=0
REPTW:    LDA  SEQDPY,X        ;Get character
          JSR  WLCD            ;Display it
          INC  X               ;
          CPX  £8              ;8 chrs yet
          BCS  REPTW           ;No keep going
          RTS                  ;Return. LCD cusor left at next
                               ;position.
;-----------------------------------------------------
RAMFUL:   CLR  X               ;X=0
REPTX:    LDA  RAMF,X          ;Get character
          JSR  WLCD            ;Display it
          INC  X               ;
          CPX  £20             ;20 chrs yet
          BCS  REPTX           ;No keep going
          JSR  ROWB            ;2nd line
          CLR  X               ;X=0
RPTX:     LDA  RAMFL,X         ;Get character
          JSR  WLCD            ;Display it
          INC  X               ;
          CPX  £20             ;20 chrs yet
          BCS  RPTX            ;No keep going
          RTS                  ;Return
```

```
;-----------------------------------------

WRNMSG:   CLR  X
RPTZ:     LDA  WRNMESG,X ;Get character
          JSR  WLCD      ;Display it
          INC  X         ;
          CPX  £20       ;20 chrs yet
          BCS  RPTZ      ;No keep going
          RTS            ;Return

;*****************************************************
;*        Timer Subroutines                          *
;*****************************************************

TIMEINIT: LDA  £$F0      ;Initialise TISR to give a 1mS
          STA  TICL      ;time delay
          LDA  £$01      ;Hex $01F4-4 =$01F0
          STA  TICH      ;Dec  0500-4  =0496
                         ;eg, 500 x 2uS=1ms
          LDA  TCRH      ;This code gets the
          STA  TEMP1H    ;contents of free
          LDA  TCRL      ;running counter
          STA  TEMP1L    ;stores it in a tempory
          CLC            ;location, adds contents of
          LDA  TICL      ;TICL & TICH to it, then stores
          ADC  TEMP1L    ;it back into the Output
          STA  TEMP2L    ;Compare Register
          LDA  TICH      ;
          ADC  TEMP1H    ;
          STA  TOCRH     ;
          LDA  TEMP2L    ;
          STA  TOCRL     ;
STCLK:    CLR  MSEC      ;Zero milliseconds
          CLR  HSEC      ;Zero hunthseconds
          LDA  £$40      ;Enable Bit 6 for interupt
          STA  TCR       ;
          LDA  TSR       ;Clear Flags
          LDA  TOCRL     ;
          CLI            ;Clear processor interupt
          RTS            ;Return

;*****************************************************
;*        Timer Interupt Service Routine             *
;*****************************************************

TISR:     SEI            ;



          LDA  TCRH      ;Interupt Service Routine
          STA  TEMP1H    ;Gets the current value of
          LDA  TCRL      ;timer counter, adds TICL & H to
          STA  TEMP1L    ;it and stores it back into
          CLC            ;timer Output Compare Reg
          LDA  TICL      ;
          ADC  TEMP1L    ;If clock = 2uS
          STA  TEMP2L    ;period of interupt
          LDA  TICH      ;= 2uS * 500
          ADC  TEMP1H    ;= 1mS
```

```
                STA   TOCRH             ,
                LDA   TEMP2L            ;
                STA   TOCRL             ;

                BRSET 6,FLAG,TOTM       ;Set for tone timer

                INC   MSEC              ;MSEC counter +1
                LDA   £100              ;
                CMP   MSEC              ;100 1mS counts yet
                BNE   RETURN            ;No then return
                CLR   MSEC              ;Yes then zero counter
                INC   HSEC              ;100mS counter
                BRA   RETURN            ;

TOTM:           DEC   LSDIGIT           ;Tone timer, counts down in 1mS
                BNE   RETURN            ;decrements to zero then sets
                DEC   MSDIGIT           ;tone end flag.
                BNE   RETURN            ;
                BSET  7,FLAG            ;Tone period finished

RETURN:         BRCLR 0,TCR,OLVL        ;
                BCLR  0,TCR             ;
                BRA   RETRN             ;
OLVL:           BSET  0,TCR             ;
RETRN:          LDA   TSR               ;Timer Flags Cleared
                LDA   TOCRL             ;
                CLI                     ;
                RTI                     ;Return From Interupt

;*************************************************************
;*                    LCD MESSAGES                           *
;*************************************************************

SYNC:           .BYTE ' <<SYNCRO - SPORT>> '
MENUE:          .BYTE 'MENU: SELECT NUMBER.'
SELECT1:        .BYTE '1:PLAY/SAVE 2:REPLAY'
SELECT2:        .BYTE '3:PLAY ONLY 4:MEMRST'
ONLYPLY:        .BYTE 'PLAY ONLY  NO STORE '
NOTE:           .BYTE 'NOTE/TIME:  '
PLYMSG:         .BYTE 'PLAY SEQUENCE: '
CLRMSG:         .BYTE '    MEMORY EMPTY    '
ZERO:           .BYTE '  32K BYTES MEMORY  '
RESET:          .BYTE '       RESET        '
FIN:            .BYTE '    RAM CLEARED     '
FALTY:          .BYTE ' 32K BYTES CHECKED  '
FNDIT:          .BYTE 'Found sequence No '
MESGA:          .BYTE 'TESTING 32K BYTE RAM'
MESGB:          .BYTE 'Finished testing RAM'
SEQDPY:         .BYTE 'SEQ No: '
RAMF:           .BYTE '<<<   WARNING    >>>'
RAMFL:          .BYTE '<  RAM  IS  FULL   >'
WRNMESG:        .BYTE '<<<  RAM FAULTY  >>>'
```

```
;*****************************************************
;*          TABLES & CONSTANTS                       *
;*****************************************************

KEYTBL:     FCB   $00,'D',$0C,$27,$18,$E4
            FCB   $01,'E',$0C,$E0,$19,$C0
            FCB   $02,'F',$0D,$A4,$1B,$48
            FCB   $03,'F',$0E,$74,$1C,$E8
            FCB   $04,'G',$0F,$50,$1E,$A0
            FCB   $05,'G',$10,$39,$20,$72
            FCB   $08,'A',$11,$30,$22,$60
            FCB   $09,'A',$11,$6E,$22,$DC
            FCB   $0A,'B',$13,$4A,$26,$94
            FCB   $0B,'C',$14,$70,$28,$E0
            FCB   $0C,'C',$15,$A8,$2B,$50
            FCB   $0D,'D',$16,$F2,$2D,$E4
            FCB   $07,'R',$00,$00,$00,$00

SEQTBL:     FCB   $00,'1'
            FCB   $01,'2'
            FCB   $02,'3'
            FCB   $03,'4'
            FCB   $04,'5'
            FCB   $05,'6'
            FCB   $08,'7'
            FCB   $09,'8'
            FCB   $0A,'9'

TIMTBL:     FCB   $00,$01,'1'
            FCB   $01,$02,'2'
            FCB   $02,$03,'3'
            FCB   $03,$04,'4'
            FCB   $04,$05,'5'
            FCB   $08,$06,'6'
            FCB   $09,$07,'7'
            FCB   $0A,$08,'8'
            FCB   $0B,$09,'9'
            FCB   $0C,$00,'0'

TONETBL:    FCB   $0C,$27    ;311.1 Hz
            FCB   $0D,$A4    ;349.2 Hz
            FCB   $0F,$50    ;392.0 Hz
            FCB   $11,$30    ;440.0 Hz
            FCB   $13,$4A    ;493.8 Hz
            FCB   $15,$A8    ;554.4 Hz
            FCB   $18,$4E    ;622.2 Hz
            FCB   $1B,$48    ;1B4.8 Hz
            FCB   $1E,$A0    ;784.0 Hz
            FCB   $22,$60    ;880.0 Hz

            END
```

The software and hardware, as described, are subject to modification as may be necessary to adapt the training device to a variety of other athletic functions which have not been described specifically in this application. Other changes and modifications will be apparent to persons skilled in the art and may be made without departing from the broad concepts of the invention as herein described and claimed.

CLAIMS

1. A sports training device to provide synchronisation signals to induce and guide movements of a sportsperson engaged in a sporting activity comprising a digital logic computer and a tone generator, the computer logic being programmed to activate the tone generator in accordance with stimulus parameters, means to input into the computer predetermined stimulus parameters based upon a behavioural analysis of models of relevant movement sequences of the sporting activity to cause the tone generator to generate a sequence of auditory pulses having predetermined characteristics and audio output means through which the generated sounds are relayed to the sportsperson as a preview and guide to the sporting activity.

2. A device as claimed in Claim 1, wherein the sequence of auditory pulses signal the onset of specific movements to be performed by the sportsperson.

3. A device as claimed in Claim 1, wherein the sequence of auditory pulses signal the onset and duration of specific movements to be performed by the sportsperson.

4. A device as claimed in Claim 1, wherein the predetermined characteristics of the pulses relate to movements of different parts of the body.

5. A device as claimed in Claim 4, wherein the predetermined characteristics also signal additional information concerning the movements to be performed.

6. A device as claimed in Claim 1, wherein said audio output means includes an earpiece to be worn by the sportsperson.

7. A device as claimed in Claim 1, wherein the audio output means includes a radio link.

8. A device as claimed in Claim 1, wherein the means to input stimulus parameters includes a keyboard.

9. A device as claimed in Claim 1, including means whereby a number of different auditory sequences may be stored concurrently.

10. A device as claimed in Claim 1, wherein means are included to store a plurality of programmes.

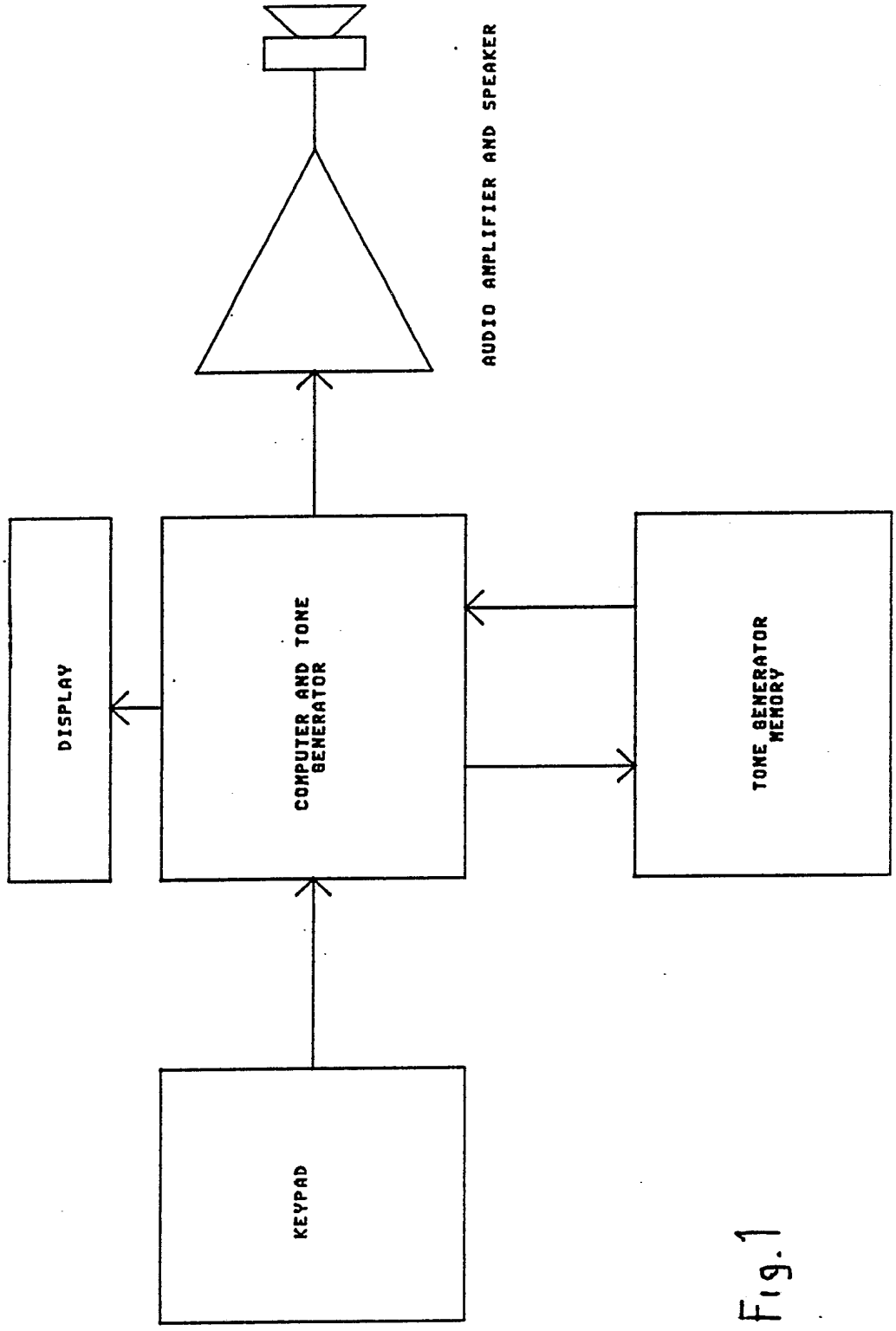

AUDIO AMPLIFIER AND SPEAKER
DISPLAY
COMPUTER AND TONE GENERATOR
TONE GENERATOR MEMORY
KEYPAD

Fig. 1

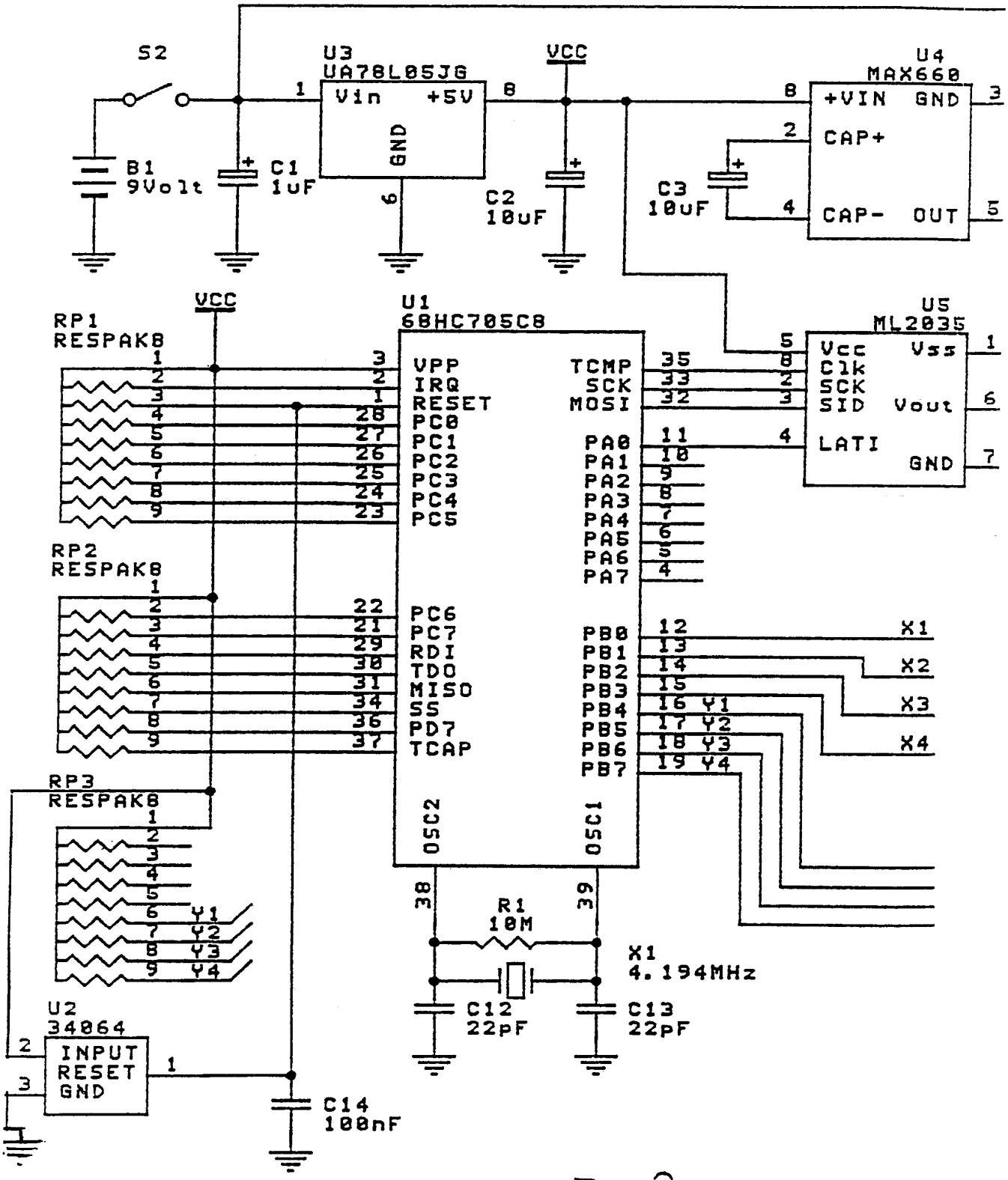

S2
U3
UA78L05JG
Vin
+5V
GND
VCC
U4
MAX660
+VIN
GND
CAP+
CAP-
OUT
B1
9Volt
C1
1uF
C2
10uF
C3
10uF
U1
68HC705C8
U5
ML2035
Vcc
Vss
Clk
SCK
SID
Vout
LATI
RP1
RESPAK8
RP2
RESPAK8
RP3
RESPAK8
VPP
IRQ
RESET
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
RDI
TDO
MISO
SS
PD7
TCAP
TCMP
SCK
MOSI
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
OSC2
OSC1
X1
X2
X3
X4
Y1
Y2
Y3
Y4
R1
10M
X1
4.194MHz
C12
22pF
C13
22pF
U2
34064
INPUT
RESET
GND
C14
100nF

Fig.2a

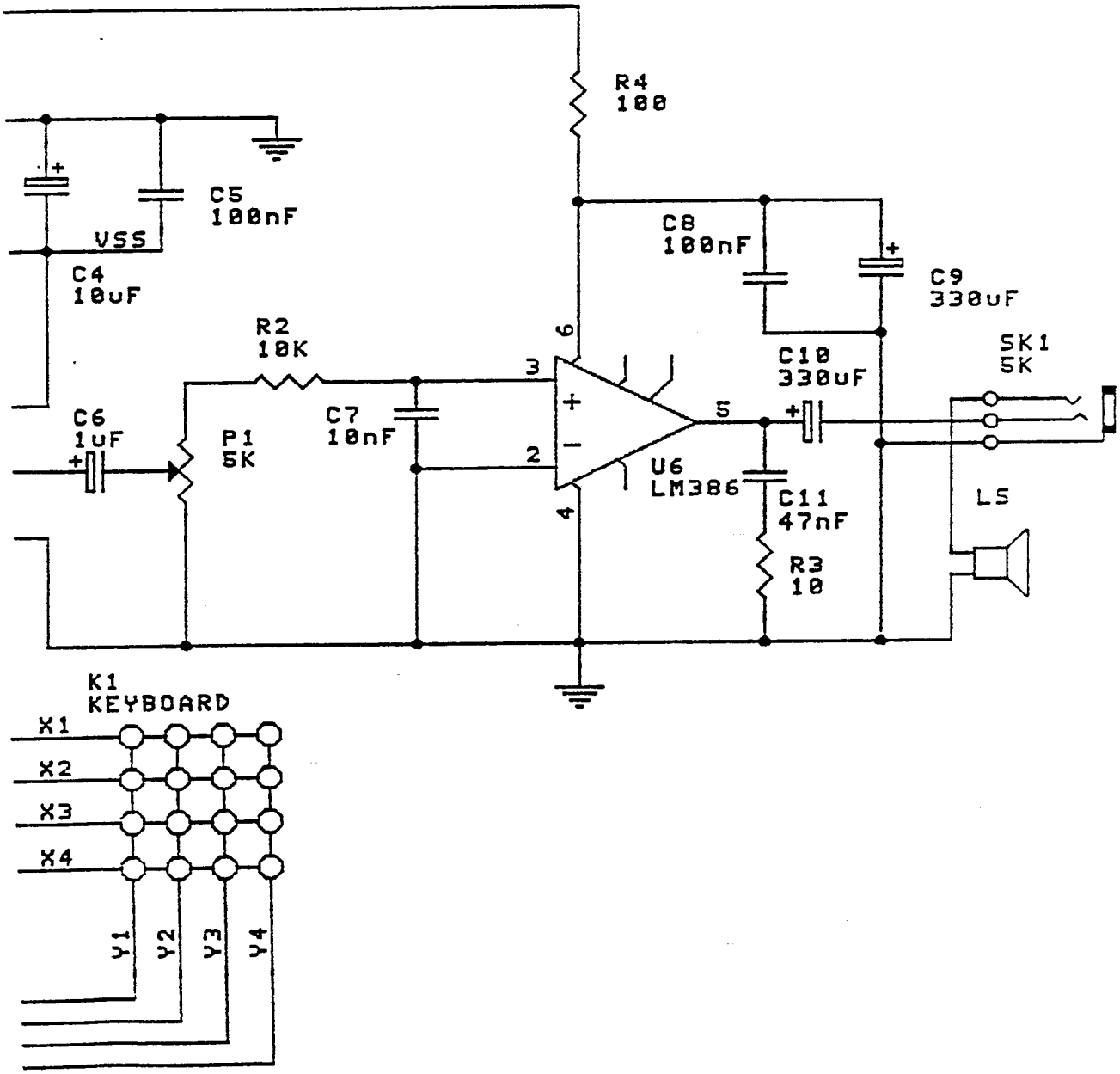

R4
100
C5
100nF
VSS
C4
10uF
C8
100nF
C9
330uF
R2
10K
C6
1uF
P1
5K
C7
10nF
U6
LM386
C10
330uF
SK1
5K
C11
47nF
R3
10
LS
K1
KEYBOARD
X1
X2
X3
X4
Y1
Y2
Y3
Y4

Fig. 2b

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU92/00237

A. CLASSIFICATION OF SUBJECT MATTER
Int. Cl.[5] G07C 1/22 A63B 26/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC G07C 1/22, A63B 24/00, 26/00, 69/00, 71/06, G04F 5/00, G06F 15/42

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
AU: IPC as above

Electronic data base consulted during the international search (name of data base, and where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate of the relevant passages	Relevant to Claim No.
A	WO,A, 89/04513 (BARJON, F) 18 May 1989 (18.05.89)	
A	DE,A, 3445654 (REINFRANK, V) 19 June 1986 (19.06.86)	
A	FR,A, 2470404 (ALLAIN, R.J.P) 5 June 1981 (05.06.81)	
A	PATENT ABSTRACTS OF JAPAN, E-149, page 86, JP,A, 53-17632 (HITACHI SEISAKUSHO K.K.) 30 August 1979 (30.08.79)	

☐ Further documents are listed in the continuation of Box C.

☐ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
1 SEPTEMBER 1992	11 Sept 1992 (11.09.92)

AUSTRALIAN PATENT OFFICE
PO BOX 200
WODEN ACT 2606
AUSTRALIA

Facsimile No. 06 2853929

Name and mailing address of the ISA/ Authorized officer

J.W. Thomson

Telephone No. (06) 2832214

John Thomson

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate of the relevant passages	Relevant to Claim No.
A	DERWENT ABSTRACT ACCESSION NO. 89-276806, CLASS T05, SU,A, 1467561 (KALININ B P) 23 March 1989 (23.03.89)	

INTERNATIONAL SEARCH REPORT
information on patent family mem. .s

International application No.
PCT/AU 92/

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report	Patent Family Member						
WO	8904513	ES EP	2010850 349613	FR	2622994	AU	27131/88